Newton-Cotes Quadrature Formulas

Chuck Allison
CNS 3320 – Numerical Software Engineering
Utah Valley State College

March 2006

Abstract

This note develops the Newton-Cotes formulas for integrating functions with evenly-spaced x-values, as well as the error analysis for Simpson's rule.

Finding areas under curves can be approximated by using interpolating polynomials for portions of the curve. The formulas for evenly-spaced partitions, known as Newton-Cotes formulas, turn out to be remarkably simple, and hence lend themselves to efficient automatic computation.

Newton-Cotes Formulas

Consider using horizontal lines (zero-degree polynomials) to approximate sections of the area under a curve. This leads to the midpoint rule, where the area between x_i and $x_i + h$ is approximately:

$$hf(x_i + \frac{h}{2})$$

Using a 1-degree polynomial connecting the points $(x_i, f(x_i)), (x_i + h, f(x_i + h))$ leads to the Trapezoidal Rule, which uses the following approximation for the area of a panel of width h:

$$\frac{h}{2}(f(x_i) + f(x_i + h))$$

Another way to find the Trapezoidal Rule is to notice that it is of the form

$$af(x_i) + bf(x_i + h)$$

In other words, it is a linear combination of the y-values $f(x_i)$ and $f(x_i + h)$. To find the values of a and b (yes, we already know they're both $\frac{h}{2}$; keep reading), we can use the fact that the functions $f_0(x) = 1$ and $f_1(x) = x$ match their interpolating polynomial of degree 1 exactly. We can then use the exact integrals for these functions as values for determining a and b. Noting that the Trapezoidal Rule is the same no matter what the x_i are, we can, without loss of generality, assume that $x_i = 0$ to obtain the following:

$$\begin{aligned} af_0(0) + bf_0(h) = a \cdot 1 + b \cdot 1 &= \int_0^h 1 \cdot dx = h \\ af_1(0) + bf_1(h) = a \cdot 0 + b \cdot h &= \int_0^h x \cdot dx = \frac{h^2}{2} \end{aligned}$$

The solution of this system of equations is $a = b = \frac{h}{2}$. So we end up with the Trapezoidal Rule by solving a system of equations for the weights of a two-point rule.

This formula is for only one panel of course. When adding up the areas of n adjacent trapezoids on the interval $[x_0, x_n]$, the *composite* formula becomes:

$$\frac{h}{2}(f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n))$$

because each of the interior $f(x_i)$ is used twice.

Simpson's Rule uses three points, which in effect uses the interpolating polynomial of degree two between the points as an approximation for the integral of $f(x)$. Proceeding as before, we integrate on the interval $[-h, h]$ three convenient functions that match the second-degree interpolating polynomial exactly ($f_0(x) = 1, f_1(x) = x, f_2(x) = x^2$), and solve the following system:

$$\begin{aligned} af_0(-h) + bf_0(0) + cf_0(h) = a + b + c &= \int_{-h}^{h} 1 \cdot dx = 2h \\ af_1(-h) + bf_1(0) + cf_1(h) = -ha + hc &= \int_{-h}^{h} x \cdot dx = 0 \\ af_2(-h) + bf_2(0) + cf_2(h) = h^2a + h^2c &= \int_{-h}^{h} x^2 \cdot dx = \frac{2}{3}h^3 \end{aligned}$$

This system of equations has solution $(a, b, c) = (\frac{h}{3}, \frac{4h}{3}, \frac{h}{3})$, so the Newton-Cotes formula for any three consecutive, evenly-spaced points, x_0, x_1, x_2 (aka Simpson's Rule), is

$$\frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2)).$$

When we sum up all the interior panels over the range $[x_0, x_n]$, we get the composite formula

$$\frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)).$$

It just so happens that $\int_{-h}^{h} x^3 dx = 0$ (of course; it's an odd function), and the Simpson's estimate for x^3 is $\frac{h}{3}(-h^3 + 0 + h^3) = 0$, so the formula is exact for degree 3 also! This always happens for even-degree interpolating polynomials. That's why we always use odd-point rules: they use even-degree interpolating polynomials and get extra accuracy for free! This will be confirmed in the next section.

Truncation Error

Taylor's Theorem for $f(x)$ expanded about x_0 is

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + f''(x_0)\frac{(x - x_0)^2}{2} + f'''(x_0)\frac{(x - x_0)^3}{3!} + f^{(4)}(x_0)\frac{(x - x_0)^4}{4!} + \ldots$$

Letting $x = x_1$ in the expansion, and setting $h = x_1 - x_0$ we get

$$f(x_1) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \frac{h^4}{4!}f^{(4)}(x_0) + \ldots$$

Similarly, letting $x = x_2$ and setting $2h = x_2 - x_0$ we get

$$f(x_2) = f(x_0) + 2hf'(x_0) + \frac{(2h)^2}{2}f''(x_0) + \frac{(2h)^3}{3!}f'''(x_0) + \frac{(2h)^4}{4!}f^{(4)}(x_0) + \dots$$

In terms of the corresponding Taylor series, the Simpson formula $A = \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2))$ becomes

$$A = \frac{h}{3}(6f(x_0) + 6hf'(x_0) + 4h^2f''(x_0) + 2h^3f'''(x_0) + \frac{5}{6}h^4f^{(4)}(x_0) + \dots$$
$$= 2hf(x_0) + 2h^2f'(x_0) + \frac{4}{3}h^3f''(x_0) + \frac{2}{3}h^4f'''(x_0) + \frac{5}{18}h^5f^{(4)}(x_0) + \dots$$

We will now determine the expansion for the true value of $I = \int_{x_0}^{x_2} f(x)$ and subtract to find an estimate for the error of the Simpson formula.

Suppose $F(x)$ is the antiderivative of $f(x)$. Then by the Fundamental Theorem of Calculus, the integral, I, is the difference $F(x_2) - F(x_0)$. Proceeding as before, we expand $F(x_2)$ as

$$F(x_2) = F(x_0) + 2hF'(x_0) + \frac{(2h)^2}{2}F''(x_0) + \frac{(2h)^3}{3!}F'''(x_0) + \frac{(2h)^4}{4!}F^{(4)}(x_0) + \frac{(2h)^5}{5!}F^{(5)}(x_0) + \dots$$

Recalling that $F'(x) = f(x)$, we obtain

$$I = F(x_2) - F(x_0) = 2hf(x_0) + \frac{(2h)^2}{2}f'(x_0) + \frac{(2h)^3}{3!}f''(x_0) + \frac{(2h)^4}{4!}f'''(x_0) + \frac{(2h)^5}{5!}f^{(4)}(x_0) + \dots$$
$$= 2hf(x_0) + 2h^2f'(x_0) + \frac{4}{3}h^3f''(x_0) + \frac{2}{3}h^4f'''(x_0) + \frac{4}{15}h^5f^{(4)}(x_0) + \dots$$

The truncation error of the Simpson formula is, of course, $I - A$. Notice how the first four terms cancel, leaving $I - A = -\frac{1}{90}h^5f^{(4)}(x_0) + \dots$. We will just write this as $I - A = O(h^5)$. This is the *local* truncation error of one Simpson evaluation over an interval of size $2h$. To approximate the *global* truncation error over the entire interval $[a, b]$, we multiply by $\frac{n}{2}$, the number of Simpson's evaluations altogether (remember that $h = \frac{b-a}{n}$):

$$\frac{n}{2}O(h^5) = \frac{nh}{2}O(h^4) = \frac{b-a}{2}O(h^4) = O(h^4).$$

(In general, if n is the degree of the interpolating polynomial approximating $f(x)$ locally, then the truncation error is $O(h^{2(1+\lfloor \frac{n}{2} \rfloor)})$).

Now let A_1 be the value of the Simpson iterate over some interval $[a, b]$, and let A_2 be the value of the Simpson iterate over $[a, b]$ with twice as many panels. In other words, the "h" for A_1 is h, and for A_2 it is $\frac{h}{2}$. We can compare the respective truncation errors as follows:

$$E_1 = I - A_1 = O(h^4)$$

$$E_2 = I - A_2 = O((\frac{h}{2})^4)$$

$$\Rightarrow E_1 \approx 16E_2$$

So doubling the density of the panels decreases the global truncation error by a factor of 16. We can now solve for I in terms of A_1 and A_2:

$$I = A_1 + E_1 \approx A_1 + 16E_2$$

but since $I = A_2 + E_2$, we can equate this to $A_1 + 16E_2$ to get

$$\Rightarrow E_2 \approx \frac{A_2 - A_1}{15}$$

$$\Rightarrow I = A_2 + E_2 \approx A_2 + \frac{A_2 - A_1}{15}$$

We have expressed E_2 in terms of A_1 and A_2, allowing us to use $E_2 = \frac{A_2 - A_1}{15}$ as a correction factor to improve A_2. E_2 is our approximation for the absolute error of the automatic integration (i.e., we continue until $E_2 \leq$ the given error tolerance—machine epsilon will not be used here, since the number of computations is so large that roundoff will likely not allow us to get full "machine accuracy"). If the user has requested an absolute error tolerance of `tol` for the integral on the entire interval $[a, b]$, then when integrating over a subset of $[a, b]$, you should use a commensurate proportion of `tol` for that subinterval. All of the above suggest the following recursive algorithm:

```
area(f, a, b, tol) {
    compute A1 and A2
    if |A2 - A1|/15 <= tol
        return A2 + (A2 - A1)/15
    else
        return area(f, a, (a+b)/2, tol/2) + area(f,(a+b)/2,b,tol/2)
}
```

Notice that this is an *adaptive* algorithm. It refines whenever the current interval does not pass the error test, but only then. If the current refinement is sufficient, we're done with the current subinterval. Hence we maximize accuracy and minimize effort.